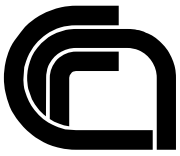

Consiglio Nazionale delle Ricerche

Formalizing Knowledge by Ontologies: OWL and KIF

A. Marchetti, F. Ronzano, M. Tesconi , S. Minutoli

IIT TR-07/2008

Technical report

Maggio 2008

Istituto di Informatica e Telematica

Formalizing Knowledge by Ontologies: OWL and KIF

Marchetti Andrea, Ronzano Francesco, Tesconi Maurizio, Minutoli Salvatore

CNR, IIT Department, Via Moruzzi 1, I-56124, Pisa, Italy
(andrea.marchetti, francesco.ronzano, maurizio.tesconi, salvatore.minutoli)@iit.cnr.it

Abstract. During the last years, the activities of knowledge formalization and sharing useful to allow for semantically enabled management of information have been attracting growing attention, expecially in distributed environments like the Web.
In this report, after a general introduction about the basis of knowledge abstraction and its formalization through ontologies, we briefly present a list of relevant formal languages used to represent knowledge: CycL, F-Logic, LOOM, KIF, Ontolingua, RDF(S) and OWL. Then we focus our attention on the Web Ontology Language (OWL) and the Knowledge Interchange Format (KIF).
OWL is the main language used to describe and share ontologies over the Web: there are three OWL sublanguages with a growing degree of expressiveness. We describe its structure as well as the way it is used in order to reasons over asserted knowledge. Moreover we briefly present three relevant OWL ontology editors: Protégé, SWOOP and Ontotrack and two important OWL reasoners: Pellet and FACT++.
KIF is mainly a standard to describe knowledge among different computer systems so as to facilitate its exchange. We describe the main elements of KIF syntax; we also consider Sigma, an environment for creating, testing, modifying, and performing inference with KIF ontologies.
We comment some meaningful example of both OWL and KIF ontologies and, in conclusion, we compare their main expresive features.

Table of Contents

List of Figures

1 Introduction

Ontologies are **formal and explicit specifications of a shared conceptualization**. They are mainly composed by: a set of *concepts* or *classes* that characterize the formalized knowledge, a set of *rules*, called also *properties* or *relations* between concepts and a set of *instances* or *individuals* belonging to the classes along with their specific properties. The individuals of a class may be characterized by a proper or not proper subset of all the relations of that class. In some way, a concept is the characterization of a set of individuals and a rule is a kind of relation that could hold between two individuals.

In general, ontologies are used to formally express knowledge about a defined domain of interest. When we want to formalize knowledge we must deal with three different levels of knowledge abstraction (see Figure 1); they must all be specified in order to provide an effective description of the information to be represented. The highest level of knowledge abstraction is the **methodological knowledge**: it is composed by all the knowledge representation languages or ontology languages like OWL or KIF that, based on a particular knowledge description formalism, provide expressive means to describe a set of classes along with their relations and constraints. All the specific sets of classes and relations constituting an ontology, defined referring to a particular ontology language and describing the general structure of a domain of interest belong to the level of **conceptual knowledge** (i.e., the Suggested Upper Merged Ontology). The set of individuals described as instances of the classes of a particular ontology, along with the relations holding between couples of them is referred to as **factual knowledge**.

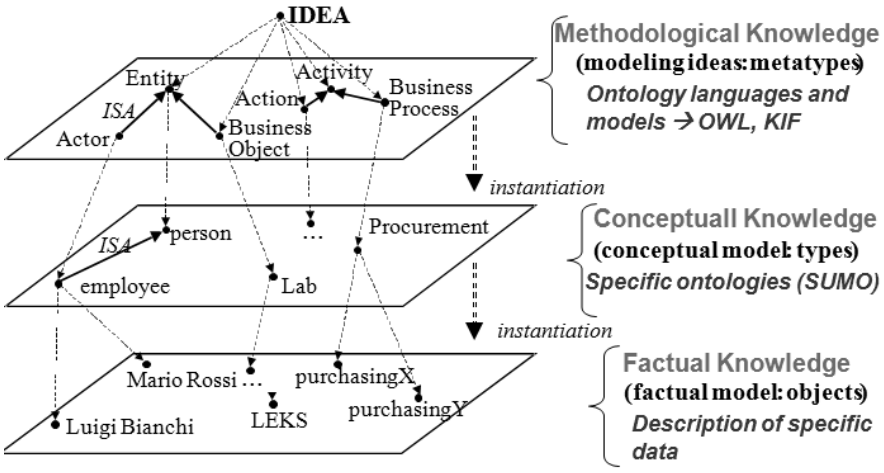

Fig. 1. The three levels of knowledge abstraction.

As said, in order to express knowledge and to make it computable, we need some sort of **formalism** that, supporting the specification of one or more ontology languages, allows for a standardized way to express, through ontologies, the information considered, making possible automated reasoning procedures. Ontologies can be expressed adopting different formalisms, called also description languages. When we choose a formalism we have to determine the right trade-off between two main opposite needs: *Expressive power* and *Complexity of reasoning*.

Description logics are a family of *knowledge representation formalisms*; they are a decidable subset of the First Order Logic (FOL). The different description logics are distinguished by different sets of constructors of concepts (union, intersection, universal and existential quantifier, etc.) and rules (inverse rule, transitive rule, concepts subsumptions, etc.). *Constructors* are the distinct expressive means available to specify *concepts* (or classes) and *rules* (or properties). The set of all the descriptions of classes and relations defines the general structure of the domain of interest along with all its constraints. It constitutes a frame of reference exploited to characterize the concrete data, that are the individuals of the considered domain along with their relations. Two widespread knowledge description languages based on description logics are the Web Ontology Language (OWL) and the Knowledge Interchange Format (KIF).

In this report, first of all we give a brief and synthetic overview of the most important knowledge representation languages available, referring Web sites to search for further information. Then we focus our attention mainly on OWL and KIF. We describe them considering their purpose, their constructs as well as their usage and the tools adopted to edit and share ontologies.

2 Overview of Semantic Description Languages

In this section we present an exhaustive list of relevant formal languages used to express concepts terms and descriptions [22].

2.1 CycL

http://www.cyc.com/cycdoc/ref/cycl-syntax.html - CycL was developed by Cycorp and it's it is a declarative language based on classical first-order logic. CycL is used to express common sense knowledge and to represent the knowledge stored in the Cyc Knowledge Base. It has six expression types: Constants, Formulas and Truth-function, Function-denotional, Variables and Quantifiers. CycL's is characterized by good expressiveness, precision, meaning and use-neutral representation. It is part of the Cyc project [1], aiming at assembling a comprehensive ontology and database of everyday common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning. CycL is used to represent the knowledge stored in the Cyc Knowledge Base (the Cyc Knowledge Base), available from Cycorp. The source code written in CycL is licensed as open source, to increase its usefulness in supporting the semantic web.

2.2 F-Logic

http://www.cs.umbc.edu/771/papers/flogic.pdf - F-Logic was developted in 1995 at Karlsruhe University - Germany and it's a formalism to represent knowledge. F-logic stands in the same relationship to object-oriented programming as classical predicate calculus stands to relational database programming. Features include, among others, object identity, complex objects, inheritance, polymorphism, query methods, encapsulation [4]. F-Logic major strengths are extensibility and his capacities to directly represent fundamental concepts that come from object oriented programming and frame based languages. F-Logic makes a number of central aspects of object oriented programming to become compatible with logic paradigm. F-Logic main weakness is related tomathematical and logical concepts needed to programme in this language. F-Logic does not possess cardinality restrictions.

2.3 LOOM

http://www.isi.edu/isd/LOOM/LOOM-HOME.html - Loom knowledge representation system has been developed by the University of Southern California's Information Sciences Institute (ISI) in 1986, under DARPA sponsorship. Loom is a language and environment for constructing intelligent applications. The heart of Loom is a knowledge representation system that is used to provide deductive support for the declarative portion of the Loom language. Declarative knowledge in Loom consists of definitions, rules, facts, and default rules. A deductive engine called a classifier utilizes forward-chaining, semantic unification and object-oriented truthmaintenance technologies in order to compile the declarative knowledge into a network designed to efficiently support on-line deductive query processing. Loom implements a suite of KR functions whose use has been validated by the substantial Loom user community. Loom is a large and complex system.

2.4 KIF

http://logic.stanford.edu/kif/specification.html - It was originally created by Michael Genesereth and others participating in the DARPA Knowledge Sharing Project. There have been a number of versions of KIF, among which SUO-KIF [25] used by Adam Pease to define SUMO. Knowledge Interchange Format (KIF) is a language designed for use in the interchange of knowledge among disparate computer systems (created by different programmers, at different times, in different languages, and so forth).KIF was created to serve as a syntax for first-order logic that is easy for computers to process. It was intended as an interlingua, rather than a format for human authoring of knowledge, but it has since been more often used for that latter purpose. KIF features full semantic expressiveness. One inconvenience of this language is his computational complexity many times has been considered too high. Although the originalKIF group intended to submit to a formal standards body, that did not occur. In ordet to read a more detailed description of KIF along with its constructs, see Paragraph 4.

2.5 Ontolingua

http://www.ksl.stanford.edu/software/ontolingua - Ontolingua, created in 1992 at Stanford University, is a language based in KIF (Knowledge Interchange Format). It provides a distributed collaborative environment to browse, create, edit, modify, and use ontologies. Combines frames paradigm and first order predicates. Beyond all the languages used to represent ontologiees, Ontolingua language is the one with the biggest expressiveness. It can represent concepts, concepts taxonomies, n-ary relationships, axioms, instances and procedures. Also because of its expressiveness, Ontolingua doesn't permit reasoning.

2.6 RDF(S)

http://www.w3.org/TR/rdf-schema/ - RDF [13] stands for Resource Description Framework and is a W3C Recommendation. RDF is a graphical language used for representing information about resources on the web thus constituting a basic ontology language. Resources are described in terms of properties and property values usingRDF statements. Statements are represented as triples, consisting of a subject, predicate and object (S, P, O). RDF is written in XML and uses URIs - Unique Resource Identifiers to identify resources. RDF Schema, along with RDF, provides basic capabilities for describing vocabularies that describe resources leaving however a lot of possibilities of extension through other important features. For a more detailed description of RDFS see Paragraph 3.

2.7 OWL

http://www.w3.org/TR/owl-features/ - Latest standard in ontology languages from the World Wide Web Consortium (W3C). OWL semantically extends RDF (S). It is based on its predecessor language DAML+OIL. OWL is an ontology language. Classes and relations are the basic building blocks of an OWL ontology. OWL has a rich set of modelling constructors. In order to allow usability by various users, OWL provides three increasingly expressivesublanguages: OWL-Lite, OWL-DL and OWL-Full. For a detailed description of OWL, its syntax and the tools that support the definition of OWL ontologies see Paragraph 3.

3 Web Ontology Language (OWL)

The Web Ontology Language (OWL) [9] is used to describe ontologies over the Web; it is intended to be a reference to specify, share and reuse processable knowledge in a distributed environment. It is built on the Resource Description Framework (RDF) [13] and RDF Schema (RDFS) [14] and provides additional vocabulary for describing properties and classes. RDF, as briefly mentioned in Paragraph 2, is a language representing information about resources over the Web. In particular in RDF each piece of information is represented as a triple composed by a property connecting two resources: the first one is

referred to as the *subject* and the second one as the *object* (ie: *subject*:Claudia - *property*:isSisterOf - *object*:Miriam). RDF Schema (metodological knowledge) provides basic constructs to define an ontology (conceptual knowledge) in order to specify RDF real data (factual knowledge); in particular it allows to define classes, properties and their subsumption hierarchies along with the domain and the range of each property. OWL was born from the need to extend RDFS to increase its expressivity, thus adding a consistent number of constructs useful to better formalize a domain.

OWL has been derived from DAML+OIL [2], an older semantic markup language for Web resources. The 1.0 version of OWL has been standardized at the beginning of 2004 as the outcome of the W3C Web Ontology Working Group. During the last few years has increased the need to extend OWL so as to add a useful set of features that have been requested by users, for which are now available effective reasoning algorithms, and that OWL tool developers are willing to support. After many proposal of extensions to OWL, since September 2007 the W3C OWL Working Group [21] has been constituted in order to formalize all these requests for extensions to produce a new standard: OWL 2.0. Currently, OWL 1.0 is mainly used along with some particular extension supported by tools developers that is likely to be standardize in OWL 2.0.

OWL formalism is based on the SHOIN(D) [3] description logic family. In particular, three different OWL sublanguages has been defined, with a growing degree of expressive power (see Figure 2):

- **OWL Lite**: it provides only simple constructs to describe domains (cardinality restrictions, optional or required properties, etc.);
- **OWL DL**: it is based on the expressive power of the SHOIN(D) description logic; it is decidable, that is that exists an algorithm which compute from the stated knowledge, the entailed knowledge in a finite number of steps;
- **OWL Full**: it adds further expressive power to OWL DL but is no longer decidable.

Nowadays the great part of OWL ontologies over the Web is expressed using OWL DL; OWL Lite is not so less expressive than OWL DL, so people usually choose OWL DL. On the other side, OWL Full is not decidable and thus standard automatic reasoning techniques cant be applied.

In what follows we give a brief overview of the main constructs of OWL DL. We contextually refer to the corresponding elements of the OWL XML presentation syntax for those constructs added by OWL 1.0 to RDFS; we also list the XML elements corresponding to the native constructors of RDFS. Those elements are respectively collected in the following namespaces: *http://www.w3.org/2002/07/owl* which is referred by the abbreviation owl and '*http://ww.w3.org/2001/01/rdf-schema*' which is referred by the abbreviation 'rdfs'.

Concept constructors:

- *union of concepts* (owl:UnionOf)
- *intersection of concepts* (owl:IntersectionOf)

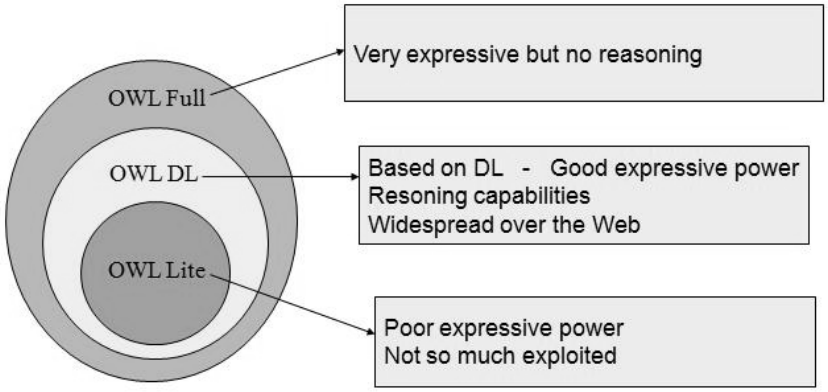

Fig. 2. The three OWL sublanguages.

- *negation of concepts* (owl:ComplementOf)
- *choice of one among more concepts* (owl:OneOf)
- *universal quantifier* (owl:AallValuesFrom)
- *existential quantifier* (owl:SomeValuesFrom)
- *greater or equal cardinality constraint between one concept and another linked through a particula property* (owl:MinCardinality)
- *less or equal cardinality constraint between one concept and another linked through a particula property* (owl:MaxCardinality)
- *equal cardinality constraint between one concept and another linked through a particula property* (owl:Cardinality)

Rules constructors and related axioms:

- *concepts subsumption* (rdfs:SubClassOf)
- *properties subsumption* (rdfs:SubPropertyOf)
- *domain of a property* (rdfs:Domain)
- *range of a property* (rdfs:Range)
- *object property* (owl:ObjectProperty)
- *datatype property* (owl:DatatypeProperty)
- *concepts equivalence* (owl:EquivalentClasses)
- *properties equivalence* (owl:EquivalentProperties)
- *instances equivalence* (owl:SameIndividual)
- *disjunction* (owl:DisjointClasses)
- *istances difference* (owl:DifferentFrom)
- *inverse rule* (owl:IinverseOf)
- *symmetric rule* (owl:SymmetricProperty)
- *transitive rule* (owl:TransitiveProperty)

– *functional property* (owl:FunctionalProperty)
– *inverse functional property* (owl:InverseFunctionalProperty)

The 'owl:oneOf' property allows the definition of enumerated classes. The element 'owl:ontology' allows expressing all the meta-information regarding the whole ontology: the URI reference for the ontology (rdf:about), a human-readable comment of the ontology (rdfs:comment), the references to previous versions (owl:priorVersion) and the references to other ontologies to include in the existing one (owl:imports).

We briefly expose some of the future extensions to OWL 1.0 that probably will be standardized by the W3C OWL Working Group durign the next years. All the proposed extensions to OWL 1.0 *keep decidability and implementability*; many of them are derived from the developments of description logic languages and reasoning techniques that have been achieved since the standardization of OWL 1.0, in 2004. First of all, some *syntactic facilities* needs to be introduced: the possibility not to define only pairwise disjoint classes but to specify *a group of classes that are disjoint* is one of them. It makes the description of domains more concise and optimizable by reasoners. OWL 1.0 users also need the possibility to define *disjointness between properties* (two properties cannot characterize the same entity at the same time) as well as to specify irreflexive and antisymmetric properties. Moreover, it is a common requirement to *express value ranges and relationships between values* (a rectangle has the width different from height). Also the possibility to *include not semantically defined comments* is a requirement for future versions of OWL. In the future directions of improvement of OWL there are also the need to *better define an XML syntax for OWL* in order to effectively exploit XPATH and XSLT processing patterns and to give users the possibility to extend OWL syntax thanks to macros.

In order to describe individuals, or better instances of the calsses belonging to an OWL ontology of reference, along with their properties the RDF is usually adopted. In this way we can define the real knowledge to carry out automated reasoning tasks. Those tasks are performed by a reasoner on the basis of the contents of the ontology and on the factual information (factual knowledge) contained inRDF triples. Usually a reasoner, applying appropriate inferencing rules, can check if there are inconsistencies in the ontology, define properties of particular individuals or also expand the factual knowledge explicitly asserted through RDF, thus derving the inferred data. In order to query for finding useful information inside RDF data collections is usually exploited SPARQL Query Language for RDF [16]. SPARQL has been standardized as a W3C Recommendation at the beginning of 2008. It gives users the possibility to query RDF graphs, defining specific information pattern to search for. In a certain sense SPARQL is important in RDF data collections like SQL is relevant to relational databases.

To better understand how all those pieces fit together we describe the simple example shown in Figure 3. On the top box is defined and graphically represented a simple OWL ontology of 'Naturally Occurring Water Sources'. It is composed

by nine classes (NaturallyOccurringWaterSources, Steam, BodyOfWater and so on). They are linked in a subsumption hierarchy through the 'rdfs:subclass' property (one of the constructs available in OWL and derived from RDF(S) to define subsumption relations between classes). In the blue-backgrounded square there is the XML representation of some RDF knowledge. In particular we say that the individual Yangtze is a river (it is an instance of the class 'River', defined in th ontology previously described). Morover we specify two properties of this particular instance: its length (6300 kilometers) and the link to another instance of the class 'Sea' ('EastChinaSea'), through the relation 'emptiesInto'. The user can query the RDF knowledge, using SPARQL for instance, or through some engine that translates natural language queries into SPARQL ones. In this way, thanks to the support of a reasoner that allows to make inferences over data relying upon the ontology, we can try to find, if it exists, a result set. As a result, the particular document, or better the particularRDF subset of data containing sensible information respect to the query is selected and show to the user, as represented in the lower yellow box of Figure 3.

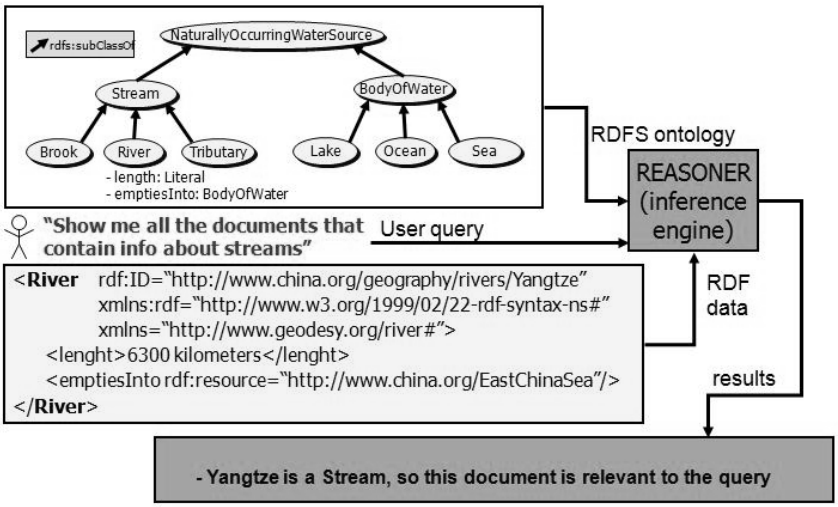

Fig. 3. Example of reasoning with an OWL ontology and RDF data.

OWL is one of the most diffused and supported languages used to describe and share ontologies over the Web; many ontologies or lexical resources are exposed exploiting OWL. As instance, the SUMO ontology has been translated into OWL [17], but also in 2006 the English 2.0 version of Wordnet lexical database has been represented by W3C in OWL and RDF [12]. In conclusion we have to mention Swoogle [18], developed by the UBMC eBiquity research group of the Department of Computer Science and Electrical Engineering of University

of Meryland, baltimore Country; it represents an interesting semantic search engine that analyzes a great amount of semantic data allowing, for instance, to search for specific classes over many indexed ontologies. It is a good resource to retrieve and explore many different OWL ontologies, in order to share, integrate and reuse conceptualizations of different domains. The great number of OWL ontologies available and the richness of their data explains the huge diffusion of OWL as a standard for ontology and knowledge description over the Web.

3.1 An examlpe of OWL ontology

In order to give a simple practical example of an OWL ontology we describe, relying on the OWL XML presentation syntax, an ontology including the classes *Person*, *Man*, *Woman* and *Father* and the property *hasChild*; OWL rules constructors like classes subsumption, classes domain and range, classes disjointness, cardinality restrictions and inverse properties are applied. All these elements are extensively commented:

```
   <?xml version=1.0?>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns
xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema
xmlns:owl=http://www.w3.org/2002/07/owl
xmlns=http://www.mylocation.it/myontology.owl
xml:base=http://www.mylocation.it/myontology.owl>


   <!-- This OWL element specifies the metadata that characterize
the ontology; in this case the empty attribute rdf:about points
out that the URI of the whole ontology is those used to refer
the file that contains it over the Web -->
<owl:Ontology rdf:about= />


   <!-- Definition of the class Person -->
<owl:Class rdf:ID=Person/>


   <!-- Definition of the class Man which is a subclass of the class
Person and its set of instances is disjoint from those of the class
Woman -->
<owl:Class rdf:ID=Man>
<rdfs:subClassOf rdf:resource=#Person/>
<owl:disjointWith rdf:resource=#Woman/>
</owl:Class>


   <!-- Definition of the class Woman which is a subclass of the
class Person and its set of instances is disjoint from those
```

```
of the class Man -->
<owl:Class rdf:ID=Woman>
<rdfs:subClassOf rdf:resource=#Person/>
<owl:disjointWith rdf:resource=#Man/>
</owl:Class>


   <!-- Definition of the class Father as a subclass of the class Man,
stating that every instance of the class father must be the subject
of at least one RDF-triple characterized by the property hasChild -->
<owl:Class rdf:ID=Father>
<rdfs:subClassOf rdf:resource=Man/>
<owl:Restriction owl:minCardinality=1 />
<owl:onProperty rdf:resource=#hasChild/>
</owl:Restriction>
</owl:Class>


   <!-- Definition of the property hasChild which must have as subject an
element/instance of the class Parent and as object an element/instance of
the class Person; its inverse property is hasParent -->
<owl:ObjectProperty rdf:ID=hasChild>

   <rdfs:domain rdf:resource=#Parent/>
<rdfs:range rdf:resource=#Person/>
<owl:inverseOf>
<owl:ObjectProperty rdf:about=#hasParent/>
</owl:inverseOf>
</owl:ObjectProperty>

</rdf:RDF>
```

In the last part of this section about OWL we will briefly describe the most important OWL editing and reasoning tools.

3.2 OWL Tools: editors and reasoners

As a consequence of OWL great diffusion, there is a huge amount of tools developed to create and edit OWL ontologies and also to reason with OWL ontologies and RDF data sets.

Among the most diffused **OWL editing tools** there are:

- **Protégé**: is an open source ontology editor developed by the Stanford Center for Biomedicl Informatic Research. It is a Java application, easily extensible thanks to a plugin mechanism. It has been adopted by a large community of users and is costantly updated and enriched with new functionalities. The Protégé-OWL extension fully supports the OWL 1.0 W3C Recommendation.

Some of the common tasks that can be carried out thanks to Protégé are: load and save OWL and RDF ontologies, edit and visualize classes, properties, and SWRL rules [20], define logical class characteristics as OWL expressions, execute reasoners such as description logic classifiers, edit OWL individuals for Semantic Web markup. To download OWL or simply to obtain more information see [11].

- **Swoop**: is a tool for creating, editing, and debugging OWL ontologies. It was produced by the MIND lab at University of Maryland, College Park, but is now an open source project with contributors from all over; it is deployed as a Java application. It has many interesting facilities to editontologies even if it is no more costantly developed. To find some more information or to download SWOOP, see [19].
- **Ontotrack**: is an integrated browser/editor of ontologies accesible as a browsing/editing system. It has many interesting interface features like sophisticate ontology layout and visualization possibilities, but it supports only OWL Lite; thus it is not possible to manage with Ontotrack the expressivity of OWL DL. To get more information about Ontotrack see [8].

Some of the most used **reasoners supporting OWL** are:

- **Pellet**: Pellet is an open source, OWL DL reasoner. It is distributed for free, but commercially supported. Pellet supports the full expressivity of OWL DL. As of version 1.4, Pellet supports many new features that has been proposed as extension for new versions of OWL, with the exception of n-ary datatypes. It is a java based web application. Pellet is widely used for reasoning tasks. To get more information about Pellet or to download it see [10].
- **FACT++**: is an OWL DL reasoner released under the GNU licence. It has been written in C++, thus maximizing performances. Beyond normal reasoning tasks, it provides some specific service like the HTML output of an OWLontologies. To download it or access to a more detailed description see [5].

4 Knowledge Interchange Format (KIF)

The Knowledge Interchange Format is **a standard to describe knowledge among different computer systems so as to facilitate its exchange**. KIF is intended not as an internal memorization format within computer, but as a mean to enable data flows among distinct systems. Itsexpressivity is based on a version of first order predicate calculus, with extensions to support non monotonic reasoning and definitions [23].

KIF, as briefly mentioned in Paragraph 2, was originally created by Micheal Genesereth and others participating in the DARPA Knowledge Sharing Effort, a global group that wanted to develop techniques, methodologies and software tools for knowledge sharing and knowledge reuse, at design, implementation, or execution time [6]. There have been a number of versions of KIF. Although

the original KIF group intended to submit to a formal standard body, that did not occur. A later version called Common Logichas since been developed for submission to ISO and has been approved and published. A variant called SUO-KIF is the language in which the Suggested Upper Merged Ontology is written [13].

In this report we refer to the version of KIF the specifications which can be retrieved at [7].

KIF has declarative semantics; this means that it is possible to understand the meaning of expressions in the language without the intermediation of any interpreter. KIF is logically comprehensive, that means that it provides for the expression of arbitrary sentences in the first order predicate calculus. Three furthercharacteristics of KIF are: the *translatability*, or better the easiness of implementation of translation mechanisms to and from particular knowledge representation languages; the *readability*, in the sense that it should be easily readable by humans even if not explicitly intended for this purpose; the *implementability*, that is the possibility, if desired, to use KIF also as a representation language within a program.

We briefly describe KIF syntax. The basic building block of KIF syntax is the character. Characters are divided into seven groups: *upper case*, *lower case*, *digits*, *alpha characters* (non alphabetical characters used in the same way letters are used), *special characters*, *spacing characters* and *other ASCII characters*. Through lexical analysis a flow of characters belonging to different groups is divided in lexemes, usually considering spacing characters as lexemes delimiters. In KIF syntax there are five types of lexemes, described in what follows. *Special lexemes* are composed by all the special characters (" - ' - # - (-) - , - \). *Words* are another type of lexeme; they are sequences of characters. Words are case insensitive and in their text, special characters are escaped through '\'. Another type of lexeme is the *Character reference*. It is composed by the characters '\' or '#' followed by any other character. They allow us to refer to characters as characters, differentiating them from one character symbols. *Character strings* are sequences of characters included in quotation marks (quotation marks are escaped by '\'). *Character blocks* allow to write a sentence of an arbitrary number of bits withot escaping; they are composed by '#' + decimal number of characters of the block + q/Q + sequence of characters. *Variables* are words in which the first character is '?' (individual variables) or '@' (sequence variables). Operators are words used to form expressions of various sort and are divided into term operators, function operators and definition operators. *Constants* are all words except variables and operators. There are object constants, used to denote individual objects, function constants, for functions on objects, relation constants, to denote relations and logical constants to express boolean conditions. Expressions in KIF are composed by one or more lexemes; according to particular rules of composition there are three types of expressions: terms, sentences and definitions. *Terms* are individual variables, character references, constants, character strings, character blocks, functional terms (function name + arguments), list term (finite list of elements),quoterm (quote operator + arbitrary list of expre-

sions) and logical terms (involving the if and the cond operators). *Sentences* are constants, equations (= operator), inequalities (\= operator), relational sentences (relation constant + arbitrary number of arguments), logical sentences (depending on the logical operator considered:conjunction, disjunction, implication, reverse implication, equivalence) and quantified sentences (existentially or universally quantified). There are three types of *definitions*: unrestricted, complete or partial. Within each type there are four classes of definitions: defobject, deffunction, defrelation, deflogical (defining respectively object, function, relation and logical constants). A *form* is a sentence or definition. A *KIF knowledge base* is a finite set of forms; the order of sentences is irrelevant. Speaking about KIF logics we must also say that:

- *functions are total* (there is a result for every combination of arguments - bottom is the undefined value);
- in functions, *list variables* (ie. @1 = 1 2 3) are considered as *multiple arguments of the same function*;
- *definitions are exploited to state sentences that are true by definition*, in a way that distingushes them from properties that express contingent properties of the world;
- *numbers are constant in base 10 representation* and there is a huge set of functions useful to elaborate them.

Considering KIF browsers and editors, we have to mention *Sigma* [16]. It has been created by Adam Pease; Sigma is an environment for creating, testing, modifying, and performing inference with ontologies. It is accessible by a browser with the support of Java libraries. As said in its presentation, Sigma shows a number of useful features for knowledge engineering work, including term and hierarchy browsing, the ability to load different files of logical theories, a full first order inference capability with structured proof results, a natural language paraphrase capability for logical axioms, support for displaying mappings to theWordNet lexicon and a numbe r of knowledge base diagnostics. In order to download the system or view the manual so as to deeply explore Sigma see [15].

4.1 An examlpe of KIF knowledge description

We comment a short example of a part of the Suggested Upper Merged Ontology (SUMO) expressed exploiting KIF. We refer to the class *Beverage*. In KIF sentences are expressed in the form: (operator/relation firstArgument secondArgument). Starting from this assumption, in line 1 we say that the class *Beverage* is a subclass of the class Food (a beverage is a particular type of food). We assume that in the previous part of SUMO there is the definition of the subclass relation. From line 2 to line 4 there is a natural language description of the class *Beverage* in English language, through the property documentation. Fro line 5 to line 7 there is an expression, involving the implication operator (=¿). It says that if there is an instance ?BEV (individual variable) of the class *Beverage* (line 6), then this instance must have as characterizing attribute the fact that it is Liquid (line 7).

```
   1. (subclass Beverage Food)
2. (documentation Beverage EnglishLanguage Äny &%Food that is ingested
3. by &%Drinking. Note that this class is disjoint with the other
4. subclasses of &%Food, i.e. &%Meat and &%FruitOrVegetable.Ÿ

5. (=>
6. (instance ?BEV Beverage)
7. (attribute ?BEV Liquid))
```

5 Conclusions

We will conclude this report exposing some consideration about OWL and KIF differences so as to try to guide the choice of the best appropriate knowledge representation language [24].

KIF is based on a set of constructs and expressive possibilities greater than OWL; to give some example of these increased descriptive possibilities we can consider that in knowledge representation languages, the context permits to represent statements over statements, also said meta-statements, and hence, for example, situation duration and statement negation, modalities, creator and argumentation relations. As instance we could want to say that: 'Laura think that Mario likes her (now) in 2003, and that before he did not'. In KIF this kind of constructs and as a consequence this kind of expressivity is possible, while in OWL 1.0 it is not. To expose a further example of differences between the two languages considered, we can state that OWL 1.0, contrary to KIF, doesn't have the possibility to define n-ary relations.

Generalizing the expressivity of OWL is not so extensive as those of KIF, but on the other side OWL is the most widespread and supported language that allows, along with RDF(S), for ontology and knowledge description, constituting the de facto standard for ontology representation over the Web and not only. This is also underlined by the great number of browsing/editing and reasoning tools developed for OWL. The huge number of OWLontologies diffused on the Web furtherly stresses the great diffusion of OWL. Moreover we must keep in mind that KIF is mainly intended as a common language to describe knowledge among different systems so as to support their exchange of data. As the last and general consideration we must say that in order to choose an ontology representation language besides all the factors just described, we must take into consideration the real need to exploit the complex descriptive capabilities of a particular language that is usually opposed to its easiness of use,reasoning and ontology editing; we must try to find the knowledge description language that better balances these two opposite needs.

References

1. The cyc project - web site. http://www.cyc.com/.
2. Daml+oil (march 2001) reference description - w3c note. http://www.w3.org/TR/daml+oil-reference.
3. Description logics from wikipedia. http://en.wikipedia.org/wiki/Description_logic.
4. F-logic description from wikipedia. http://en.wikipedia.org/wiki/F-logic.
5. Fact++ - web site. http://owl.man.ac.uk/factplusplus/.
6. Knowledge interchange format from wikipedia. http://en.wikipedia.org/wiki/Knowledge_Interchange_Format.
7. Knowledge interchange format specifications - draft proposed american national standard (dpans). http://logic.stanford.edu/kif/dpans.html.
8. Ontotrack - web site. http://www.informatik.uni-ulm.de/ki/ontotrack/.
9. Owl web ontology language overview - w3c recommendation. http://www.w3.org/TR/owl-features/.
10. Pellet - web site. http://pellet.owldl.com/.
11. Protégé-owl - web site. http://protege.stanford.edu/overview/protege-owl.html.
12. Representation of wordnet in rdf/owl - w3c working draft. http://www.w3.org/TR/wordnet-rdf/.
13. Resource description framework (rdf) - w3c recommendation. http://www.w3.org/TR/REC-rdf-syntax/.
14. Resource description framework schema (rdfs) - w3c recommendation. http://www.w3.org/TR/rdf-schema/.
15. Sigma knowledge engineering environment for kif knowledge - web site. http://sigmakee.sourceforge.net/.
16. Sparql query language for rdf - w3c recommendation. http://www.w3.org/TR/rdf-sparql-query/.
17. The suggested upper merged ontology expressed adopting owl. http://www.ontologyportal.org/translations/SUMO.owl.txt.
18. Swoogle: semantic web search - web site. http://swoogle.umbc.edu/.
19. Swoop - web site. http://code.google.com/p/swoop/.
20. Swrl: a semantic rule language combining owl and ruleml - w3c member submission. http://www.w3.org/Submission/SWRL/.
21. The w3c owl working group 2007 - web site. http://www.w3.org/2007/OWL/wiki/OWL_Working_Group.
22. Semantic domain system group of Universidade de Madeira. Semanic description languages. http://seed.uma.pt/Projects/sds/index.php?page=lang.html.
23. Labrou Finin and Mayfield. A brief introduction to the knowledge interchange format. http://www.cs.umbc.edu/kse/kif/kif101.shtml.
24. Martins. Knowledge representation/translation in rdf+owl, n3, kif, uml and the webkb-2 languages. http://www.cit.gu.edu.au/ phmartin/WebKB/doc/model/comparisons.html.
25. Adam Pease. Suo-kif - standard upper ontology knowledge interchange format. http://sigmakee.cvs.sourceforge.net/*checkout*/sigmakee/sigma/suo-kif.pdf.